**文件与结构体的操作**

一、结构体与文件

* 1. 直接上代码

//typedef struct student{

// char name[30];

// int age;

// double score;

//}Stu;

#pragma pack(1)

typedef struct student{

char name[30];

int age;

double score;

}Stu;

#pragma pack()

int main()

{

//我们打印出来看到结构体的字节是48bit，但是我们实际上才花了42个bit

//这就得谈论到内存对齐了，这与编译器是相关的，这说明我的编译器是8字节

//那什么是对齐呢？比如我的编译器是8位‘

//那么，第一个是30个字节，但是要凑齐8的倍数，所以会凑成32

//第二个是4个字节，所以会凑成8

//第三个刚好不用，所以我们一算是32+8+8 = 48

//那么问题来了，我们的编译器千差万别，那我们怎么规定内存对齐呢

//用#pragma pack(x) #pragma pack()

int stuSize = sizeof(Stu);

printf("the Stu is %d bit\n", stuSize);

//实例化结构体

Stu students[2];

//进行赋值

strcpy(students[0].name, "zhuli");

students[0].age = 22;

students[0].score = 90.0f;

strcpy(students[1].name, "weierLin");

students[1].age = 23;

students[1].score = 99.0f;

//我们现在打开文件进行写的操作

FILE \*fpw;

fpw = fopen("stuFileInfo.bin", "wb"); //以二进制进行写

if(!fpw){

printf("file open error");

exit(0);

}

int result = fwrite(students, sizeof(Stu), 2, fpw); //结构体首地址i、字节长度、个数、文件名

if(result > 0){

printf("write succeed\n");

fclose(fpw);

}

//先定义结构体来存储

Stu studentNew[2];

//那么现在我们要读出来

FILE \*fpr;

fpr = fopen("stuFileInfo.bin", "rb");

result = fread(studentNew, sizeof(Stu), 2, fpr);

if(result > 0){

printf("read success\n");

fclose(fpr);

}

//我们可以进行打印以下

Stu \*stu = studentNew;

for(int i = 0; i < 2; i++){

printf("name:%s, age:%d, score:%f\n", stu->name, stu->age, stu->score);

stu++;

}

return 0;

}